Consulta Mongo

Michael Stiven Castillo Castillo Id 825947

Corporación Universitaria Minuto de Dios

Ingeniería de Sistemas,

Bases De Datos Masivas

Alexander Matallana Porras

21 de marzo del 2025

2. Introducción

- MongoDB es una base de datos NoSQL orientada a documentos, diseñada para almacenar y gestionar grandes volúmenes de datos de forma flexible a diferencia de las bases de datos relacionales tradicionales, que usan tablas con filas y columnas, MongoDB almacena la información en documentos con una estructura similar a JSON dentro de colecciones
- Esto le permite prescindir de un esquema fijo, adaptándose fácilmente a datos semiestructurados o no estructurados. En el presente informe se explora qué es MongoDB, sus principales características técnicas y diferencias con sistemas relacionales como MySQL, haciendo énfasis en su modelo de datos de documentos, formato de almacenamiento, operaciones básicas y ventajas en escalabilidad y flexibilidad. La finalidad es brindar una comprensión clara de MongoDB y de por qué se ha convertido en una herramienta ampliamente utilizada en el ámbito de las bases de datos modernas.

- 3. Objetivos

✓ Comprender la naturaleza de MongoDB como base de datos NoSQL y diferenciarla de las bases de datos relacionales tradicionales (como MySQL) en cuanto a modelo de datos y funcionamiento.

✓ Describir la estructura de datos y operaciones fundamentales en MongoDB, incluyendo el concepto de colecciones y documentos, el formato JSON/BSON y las operaciones básicas CRUD para manipular la información.

✓ Analizar las ventajas técnicas de MongoDB en términos de flexibilidad y escalabilidad, ilustrando cómo maneja relaciones sin joins, cómo puede desplegarse (ej. con Docker) y qué herramientas existen para su administración visual.

4. Tipo de base de datos de MongoDB vs. MySQL (Relacional)

MongoDB es un sistema de gestión de bases de datos NoSQL, lo que significa que no se basa en un modelo relacional y está orientado a documentos. A diferencia de las bases de datos relacionales tradicionales como MySQL, que organizan la información en tablas estructuradas con filas y columnas predefinidas, MongoDB almacena los datos en documentos flexibles. Esto permite que no exista un esquema rígido; cada documento puede tener una estructura distinta. MySQL, siendo una base de datos relacional, impone esquemas estrictos, lo que lo convierte en una opción preferible cuando se busca una alta integridad y consistencia de datos, como en el caso de transacciones financieras. En contraste, el enfoque menos restrictivo y el mejor rendimiento de MongoDB lo hacen más adecuado para aplicaciones donde son primordiales la disponibilidad y la velocidad. En resumen, MySQL es una base de datos SQL estructurada que ofrece una fuerte consistencia, mientras que MongoDB es NoSQL, caracterizándose por su mayor flexibilidad y escalabilidad horizontal.

5. Colecciones en MongoDB vs. Tablas en SQL

En MongoDB, los datos se organizan en colecciones, que son el equivalente lógico de las tablas en las bases de datos relacionales. Cada colección agrupa un conjunto de documentos (registros) y se encuentra dentro de una base de datos específica. Un documento en MongoDB está formado por pares clave-valor y representa la unidad básica

de datos, similar a una fila en SQL. Sin embargo, a diferencia de las filas de una tabla SQL, los documentos de una colección no necesitan tener los mismos campos o estructuras. MongoDB no impone un esquema fijo, lo que significa que cada documento dentro de la misma colección puede contener campos distintos o adicionales según lo que requiera la información. Esta flexibilidad contrasta con las tablas SQL, donde todas las filas deben compartir exactamente las mismas columnas definidas. En resumen, una colección de MongoDB desempeña un papel similar al de una tabla en una base de datos relacional, pero con un esquema dinámico que permite variaciones en los documentos almacenados.

6. Almacenamiento de la información en MongoDB y formato de datos

MongoDB almacena la información en un formato de documento JSON binario, conocido como BSON (Binary JSON). Internamente, cada documento se guarda como un objeto BSON, que es la representación binaria de un documento JSON. Este formato binario permite a MongoDB manejar los datos de manera muy eficiente, aprovechando tipos de datos adicionales y optimizaciones en la lectura y escritura. Los documentos BSON se agrupan en colecciones dentro de la base de datos, y MongoDB tiene la capacidad de distribuir estos documentos en múltiples servidores para mejorar la escalabilidad.

Desde la perspectiva del desarrollador, un documento de MongoDB se asemeja a un objeto JSON clásico, con llaves {}, nombres de campo y valores. Sin embargo, al ser almacenado en la base de datos, este documento se transforma al formato BSON binario. Este modelo permite que MongoDB albergue datos estructurados, semiestructurados o no estructurados

en un mismo repositorio. En la práctica, cuando interactuamos con MongoDB, ya sea a través de la consola o de un controlador, los datos que vemos y enviamos están en un formato JSON legible. Posteriormente, el servidor realiza la conversión a BSON para almacenarlos adecuadamente.

7. JSON y BSON en MongoDB: ¿Cuál es la diferencia?

JSON, o Notación de Objetos de JavaScript, se asemeja a un documento claro y accesible donde se registran datos de manera sencilla. Por otro lado, BSON, utilizado en MongoDB, actúa como una especie de lenguaje encriptado, diseñado específicamente para el manejo de datos.

La clave aquí radica en que JSON es fácil de interpretar, como un libro abierto disponible para todos; mientras que BSON presenta un conjunto de características más complejas, similares a un enigma en código binario, destinado únicamente a las máquinas.

A pesar de esto, BSON destaca por su rapidez en el acceso a los datos y su eficiencia en el uso del espacio, ya que incluye información adicional, como el tamaño de los textos y sus estructuras. Además, soporta una gama más amplia de tipos de datos que JSON; por ejemplo, maneja fechas, números de diferentes tamaños, archivos y códigos únicos, elementos que JSON no puede gestionar.

Gracias a estas ventajas, MongoDB es capaz de almacenar y buscar datos complejos de manera más eficaz. En resumen, podríamos decir que JSON es como un idioma universal que todos comprenden, mientras que BSON se asemeja a un código interno que utiliza MongoDB. Los programadores trabajan con JSON por su simplicidad, y es MongoDB quien se encarga de la conversión a BSON. Es un proceso fascinante, casi mágico, que refleja el modo en que funciona esta tecnología.

8. Estructura de los archivos JSON

Los archivos JSON tienen una estructura similar a un árbol, organizada por pares clave-valor. La estructura básica comienza y termina con llaves `{ }`. Cada clave, que funciona como etiqueta, siempre va entre comillas, seguida por dos puntos (`:`) y luego el dato correspondiente.

Los datos pueden ser:

String = Texto	Una cadena entre comillas.	Ejemplo: "Hola, mundo".
Number = Número	Enteros o decimales sin comillas	Ejemplo: 42, 3.14.
Boolean = Booleano	Valores de verdadero o falso, sin comillas.	Ejemplo: true, false.

Null = Nulo	Ausencia de valor o información.	Ejemplo: null.
Array = Arreglo	Una lista ordenada de valores, encerrada entre corchetes [ ].	Ejemplo: [1, 2, 3].
Object = Objeto	Agrupación jerárquica de pares clave-valor dentro de llaves { }.	Ejemplo: { "nombre": "Juan", "edad": 30 }.

```json
{
  "nombre": "Juan",
  "edad": 28,
  "aficiones": ["fútbol", "ajedrez", "surf"],
  "activo": true,
  "direccion": { "ciudad": "Bogotá", "pais": "Colombia" }
}
```

En este ejemplo, "nombre" es una clave con valor string, "edad" es un número, "aficiones" es un array de strings, "activo" es booleano, y "direccion" es un objeto embebido con sus propios campos. Esta estructura muestra la flexibilidad de JSON para representar datos complejos de forma legible.

9. Qué ventajas tiene MongoDB sobre una base de datos relacional en términos de escalabilidad y flexibilidad

MongoDB ofrece varias ventajas significativas frente a las bases de datos relacionales tradicionales, especialmente en escalabilidad horizontal y flexibilidad de modelo de datos. En términos de escalabilidad, MongoDB fue concebido para escalar de forma distribuida: es capaz de crecer "hacia afuera" añadiendo más nodos servidores en lugar de depender únicamente de un servidor más potente. Implementa una técnica llamada sharding (fragmentación de datos) que permite dividir una colección de datos muy grande en múltiples fragmentos distribuidos en diferentes nodos o máquinas.

Cada fragmento (shard) contiene una parte del conjunto total de documentos, y juntos conforman una base de datos lógica unificada. Gracias a esto, MongoDB puede manejar volúmenes masivos de información y altas tasas de operaciones mediante la adición de nodos, manteniendo un buen rendimiento. Por ejemplo, empresas y aplicaciones que requieren bases de datos de gran escala pueden desplegar clusters de MongoDB con decenas o incluso cientos de nodos, repartiendo la carga de trabajo.

Este enfoque de escalamiento horizontal suele ser más rentable y fácil de administrar que los intentos de escalar verticalmente una base de datos SQL (lo cual implicaría adquirir hardware cada vez más potente, con costos crecientes y un límite físico). Además, MongoDB soporta la replicación nativa mediante conjuntos de réplicas, de modo que los datos se duplican en múltiples servidores para proporcionar alta disponibilidad y tolerancia a fallos. En una base de datos relacional tradicional, lograr una escala similar requiere arquitecturas más complejas (clústers, particionamiento manual, replicación maestro-esclavo, etc.), que suelen ser menos flexibles.

En cuanto a la flexibilidad, MongoDB sobresale porque no impone un esquema fijo para los datos. Esto significa que la estructura de los documentos puede evolucionar con el tiempo sin necesidad de migraciones formales de esquema (como sí ocurre en SQL cuando se altera una tabla). Los documentos de una misma colección pueden tener diferentes campos o estructuras según se requiera, y el sistema los admite sin problemas.

Esta flexibilidad permite desarrollar aplicaciones de forma ágil, ya que el modelo de datos puede ajustarse sobre la marcha conforme cambian los requisitos, añadiendo nuevos campos o anidando nuevos subdocumentos sin afectar a los documentos existentes. En una base relacional, tal cambio implicaría modificar el esquema de la tabla (por ejemplo, agregar columnas), lo cual puede ser costoso en entornos de producción y requerir downtime o scripts de migración de datos. MongoDB, en cambio, permite agregar un campo nuevo a algunos documentos simplemente incluyéndolo en esos documentos, mientras que otros documentos que no lo tengan siguen siendo válidos. Esta capacidad de almacenar datos semiestructurados es muy útil para conjuntos de datos heterogéneos o cuando se manejan datos de distintas fuentes con atributos variables. Por ejemplo, en una aplicación web que almacena perfiles de usuarios, algunos usuarios podrían tener un campo "biografía" y otros no; en MongoDB ambos casos coexisten en la misma colección sin problema. Esta libertad ofrece una adaptabilidad que acelera el desarrollo y facilita representar información compleja (como estructuras jerárquicas o anidadas) de manera natural.

10. Comandos para realizar operaciones CRUD en MongoDB

En MongoDB, las operaciones básicas de manejo de datos se conocen como operaciones CRUD (por Create, Read, Update, Delete, es decir, Crear, Leer, Actualizar, Eliminar). A continuación, se mencionan los principales comandos o métodos para realizar cada una de estas acciones en la interfaz de MongoDB (por ejemplo, usando la consola shell de MongoDB o los controladores en un lenguaje de programación).

- ✓ Crear (Create): Para insertar nuevos documentos en una colección se utilizan los métodos insertOne() (inserta un único documento) y insertMany() (inserta múltiples documentos a la vez) Ejemplo, en la shell uno podría ejecutar db.miColeccion.insertOne({ nombre: "Alice", edad: 25 }) para crear un documento con esos datos. MongoDB también crea la colección automáticamente si no existía cuando se inserta el primer documento.

- ✓ Leer (Read): La consulta o lectura de documentos se realiza típicamente con el método find() sobre una colección. Ejemplo, db.miColeccion.find({ edad: { $gt: 20 } }) recuperaría todos los documentos de miColección cuyo campo edad es mayor a 20. Existe también findOne() para obtener un solo documento. Estas funciones permiten especificar filtros (criterios de búsqueda) y proyectar campos (elegir qué campos devolver).

- ✓ Actualizar (Update): Para modificar documentos existentes se emplean métodos como updateOne() (actualiza el primer documento que coincide con un criterio) y updateMany() (actualiza todos los documentos que cumplen el criterio). Así mismo,

MongoDB ofrece diferentes operadores de actualización (por ejemplo, $set para asignar nuevos valores a campos, $inc para incrementar numéricamente, etc.). Un ejemplo en la shell: db.miColeccion.updateOne({ nombre: "Alice" }, { $set: { edad: 26 } }) actualizaría la edad del documento cuyo nombre es "Alice". Existe también replaceOne() que reemplaza por completo un documento.

- ✓ Eliminar (Delete): Para borrar documentos se utilizan deleteOne() (elimina un documento que cumpla el criterio dado) y deleteMany() (elimina todos los documentos que coincidan con el criterio). Ejemplo: db.miColeccion.deleteMany({ activo: false }) eliminaría todos los documentos de miColección cuyo campo activo sea false. También existía en versiones antiguas el método remove(), pero en las versiones modernas se prefiere deleteOne/Many.

Además de estos métodos, MongoDB proporciona comandos similares a nivel de su API o drivers en distintos lenguajes, pero los nombres se mantienen consistentes. Es importante destacar que antes de hacer operaciones CRUD, usualmente se selecciona la base de datos con use nombreBaseDatos (en la shell) y eventualmente se puede crear explícitamente una colección con db.createCollection("nombre") si se desea establecer opciones particulares; sin embargo, como se mencionó, la creación implícita ocurre al insertar. Los comandos CRUD de MongoDB ofrecen una sintaxis relativamente simple y expresiva, utilizando documentos JSON para los criterios de consulta y las actualizaciones, lo que resulta natural para los desarrolladores familiarizados con objetos JSON. Estas operaciones son análogas a

las de SQL (INSERT, SELECT, UPDATE, DELETE) pero adaptadas al modelo de documentos de MongoDB.

11. Cómo se pueden relacionar datos en MongoDB sin utilizar joins como en SQL

En las bases de datos relacionales como MySQL, es común establecer relaciones entre tablas (uno a muchos, muchos a muchos, etc.) y luego combinar datos relacionados mediante operaciones JOIN en las consultas SQL. MongoDB, al ser NoSQL orientado a documentos, no dispone de joins tradicionales en sus operaciones básicas de consulta. Entonces, ¿cómo se representan y utilizan las relaciones entre datos? La respuesta es que MongoDB maneja las relaciones de dos modos principales: incrustación (embebido) de documentos y referencias manuales. En lugar de hacer un join al vuelo en cada consulta, MongoDB suele favorecer que los datos relacionados se almacenen juntos o que el vínculo se resuelva a nivel de la aplicación cliente.

Embeber documentos (incrustación): Consiste en almacenar documentos anidados dentro de otros documentos para representar relaciones padre-hijo. Por ejemplo, si en una aplicación relacional tendríamos una tabla de Pedidos y otra de Items (detalles del pedido) relacionadas por una clave foránea, en MongoDB podríamos tener una colección pedidos donde cada documento de pedido contiene un arreglo de subdocumentos de items. De este modo, toda la información relacionada está en un mismo documento y se recupera en una sola consulta, eliminando la necesidad de un join.

La incrustación es ideal para relaciones uno-a-muchos donde el lado "muchos" (los subdocumentos) se accede siempre en contexto del "uno" (documento principal) y no de forma independiente.

En otros casos, mantener datos embebidos no es práctico (por ejemplo, una relación muchos-a-muchos o cuando los datos relacionados son muy grandes o compartidos). En esos escenarios, se pueden almacenar referencias manuales, típicamente utilizando el identificador único _id de un documento como referencia en otro. Por ejemplo, podríamos tener una colección estudiantes y una colección cursos, y en cada documento de estudiante almacenar un campo cursosInscritos que contenga una lista de identificadores (ObjectId) de cursos en los que está inscrito. Para obtener los datos relacionados, la aplicación primero consulta el estudiante, obtiene sus IDs de cursos, y luego realiza consultas a la colección de cursos para traer los detalles. Este enfoque requiere múltiples consultas, pero MongoDB es muy rápido manejando consultas por _id (que está indexado por defecto). Alternativamente, MongoDB ofrece la etapa de agregación $lookup que simula un join dentro del Aggregation Framework (una funcionalidad de consultas avanzadas), pero su uso es más complejo y no es tan común en operaciones sencillas.

En cualquier caso, las relaciones en MongoDB se gestionan sin joins explícitos, ya sea estructurando los datos juntos o mediante referencias que la lógica de la aplicación resuelve. En la documentación se enfatiza que en sistemas como MongoDB las relaciones se establecen mediante incrustación o referencia de datos, en lugar de joins en las consultas.

Esto implica un cambio de paradigma: se suele diseñar el modelo de datos de MongoDB tratando de agrupar datos relacionados que se consultan juntos (desnormalización controlada) para optimizar lecturas, y aceptando cierta redundancia si es necesario. Por ejemplo, en lugar de normalizar completamente todos los datos como en un modelo relacional, en MongoDB a veces se duplican campos en varios documentos para evitar un join (e.g., almacenar el nombre de categoría en cada producto en lugar de tener una tabla de categorías separada). Esa duplicación se considera aceptable si mejora el desempeño de lectura, siempre y cuando se maneje cuidadosamente la sincronización (actualizar todos los lugares donde se duplica cuando cambie). En resumen, para relacionar datos en MongoDB se prefiere embeber documentos o usar referencias y manejar la combinación desde la aplicación, lo cual difiere del modelo SQL pero aprovecha la flexibilidad y rapidez de acceso por documento de MongoDB.

12, Instrucciones para descargar una imagen de MongoDB usando Docker

El uso de Docker para desplegar MongoDB es muy común porque facilita la instalación y aislamiento de la base de datos en un contenedor. Para descargar la imagen oficial de MongoDB con Docker, se deben seguir estos pasos básicos:

Instalar Docker (si no se tiene instalado previamente). Docker está disponible para Linux, Windows y Mac; se debe instalar Docker Engine o Docker Desktop según el entorno. Verificar la instalación con docker --version.

Obtener la imagen de MongoDB desde Docker Hub. Docker Hub es el repositorio central de imágenes. MongoDB tiene una imagen oficial allí. Para descargarla, se utiliza el comando de terminal:

docker pull mongo

Este comando descargará la última versión disponible de la imagen oficial de MongoDB

Por defecto, mongo etiqueta la versión latest (más reciente estable). La imagen contiene todo lo necesario para ejecutar un servidor MongoDB en contenedores.

```
C:\Users\USER>docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
5a7813e071bf: Pull complete
d67c4ebf9460: Pull complete
7afa02f8c09e: Pull complete
4e7ca17a42bd: Pull complete
342a4f4728ff: Pull complete
d5bafd14fbe8: Pull complete
0c492c8e8cfd: Pull complete
734719e891c0: Pull complete
Digest: sha256:7bd28e5eea1c5766a084d5818254046f3ebe3b8f20a65e3a274640189e296667
Status: Downloaded newer image for mongo:latest
docker.io/library/mongo:latest

C:\Users\USER>
```

Creamos un contenedor

```
C:\Users\USER>docker run -d --name mi_mongo -p 27017:27017 mongo
ecaa137e7c9e638cf656f538032f21486983708c5cc8b2f4a90fd9cce6d22029
```

13. Herramientas gráficas para visualizar datos en MongoDB (similares a MySQL Workbench)

Al igual que MySQL cuenta con Workbench para administración visual, MongoDB dispone de diversas interfaces gráficas (GUI) que facilitan la visualización y gestión de los datos. La herramienta oficial es MongoDB Compass, una aplicación de escritorio proporcionada por los creadores de MongoDB. Compass permite conectarse a una instancia de MongoDB y explorar de forma interactiva las bases de datos, colecciones y documentos, ofreciendo una vista clara del esquema de los datos y opciones para filtrarlos, editarlos y agregarlos mediante una interfaz amigable. Es una herramienta muy completa que ayuda tanto a usuarios novatos como avanzados a manejar el lenguaje de consultas de MongoDB (MQL) de manera intuitiva, mostrando resultados de consultas y estadísticas de rendimiento de forma visual.

Robo 3T integra una consola shell de MongoDB, permitiendo ejecutar comandos directamente, a la vez que presenta los resultados estructurados en JSON. Otra opción robusta es Studio 3T, una suite comercial (con versión gratuita limitada) que provee un IDE completo para MongoDB, incluyendo un diseñador de consultas visual, soporte para SQL (traduce consultas SQL a MQL), herramientas de importación/exportación de datos y editor de agregaciones. Estas herramientas brindan funcionalidades similares: navegar por las bases de datos, editar documentos, crear colecciones y ejecutar consultas, todo mediante interfaces gráficas. En resumen, para MongoDB hay disponibles varias aplicaciones GUI que cumplen el rol análogo a Workbench de MySQL. MongoDB Compass es la opción oficial recomendada para comenzar, complementada por alternativas como Robo 3T, Studio 3T u otras (como NoSQLBooster, Humongous, Mongo Management Studio, etc.), las

cuales facilitan la administración y consulta de datos en MongoDB de forma visual e interactiva.

14. Conclusiones

MongoDB se ha consolidado como una de las bases de datos NoSQL más prominentes gracias a su enfoque orientado a documentos, que le confiere una gran versatilidad para manejar datos modernos. A lo largo de este informe, hemos visto que MongoDB almacena información en documentos BSON, equivalentes a objetos JSON flexibles, lo que elimina la rigidez de los esquemas predefinidos y permite adaptarse con facilidad a datos semi-estructurados o en evolución. Esto contrasta con las bases de datos relacionales como MySQL, donde la estructura de datos debe definirse de antemano mediante tablas y columnas. La capacidad de esquema dinámico de MongoDB es una ventaja clave en entornos donde los requisitos cambian rápidamente o los datos no tienen una forma uniforme.

15. Bibliografías

IBM. (s.f.). ¿Qué es MongoDB? IBM Topics. Recuperado el 21 de marzo de 2025, de https://www.ibm.com/es-es/topics/mongodb

DataScientest. (2021, 7 de abril). MongoDB: todo sobre la base de datos NoSQL orientada a documentos. Recuperado de https://datascientest.com/es/mongodb-todo-sobre-la-base-de-datos-nosql-orientada-a-documentos

Amazon Web Services. (s.f.). Comparación entre MongoDB y MySQL: Diferencia entre los sistemas de administración de bases de datos. AWS. Recuperado de https://aws.amazon.com/es/compare/the-difference-between-mongodb-vs-mysql/

Aula301. (s.f.). ¿Qué tipos de datos podemos utilizar en MongoDB? Recuperado de https://aula301.com/tipos-datos-podemos-utilizar-mongodb/

Cortés, D. (2022, 3 de julio). CRUD en MongoDB: Guía esencial de consultas y operaciones. Medium. Recuperado de https://medium.com/@diego.coder/consultas-y-operaciones-b\u00e1sicas-en-mongo-db-crud-operadors-77fe912776a7

Cortés, D. (s.f.). ¿Relaciones en MongoDB? ¡Sí es posible! Guía Rápida. Medium. Recuperado de https://medium.com/@diego.coder/relaciones-en-mongodb-edf2107a94ad

Equipo editorial de IONOS. (2024, 15 de julio). MongoDB Compass: instalación y primeros pasos. IONOS Digital Guide. Recuperado de https://www.ionos.com/es-us/digitalguide/paginas-web/desarrollo-web/mongodb-compass/

Pure Storage. (s.f.). ¿Qué es MongoDB y cómo funciona? Pure Storage Knowledge. Recuperado de https://www.purestorage.com/es/knowledge/what-is-mongodb.html

JoLuGaMa. (2021, 24 de mayo). MongoDB – tutorial básico – Atlas, Compass. JoLuGaMa Blog. Recuperado de https://jolugama.com/blog/2021/05/24/mongodb-tutorial-basico/